计算机视觉实验三——词袋模型

2021113117 王宇轩

1. 实验环境

- 操作系统: Windows
- 编程语言：Python

2. 文件列表

文件名	内容
sift.py	SIFT实现
main.py	主程序
实验报告.pdf	实验报告

3. 实验过程

若要运行程序，请将数据集的`raw_image`目录（其下为`train`和`test`目录）与main.py、sift.py放置于同一目录。

3.1 SIFT算法提取特征点

实现的SIFT算法详见**sift.py**文件。在main.py中，通过

```
kp, des = sift.computeKeypointsAndDescriptors(img)
```

进行调用。在main.py中，因为模型训练函数`trainModel()`和测试函数`testModel()`都需要调用，因此打包成了函数`getDescriptors()`。

3.2 生成词袋模型

在**main.py**中，通过sift算法提取到特征点及描述后，用K均值聚类的方法生成词表。这一部分**并没有直接调用现成的词表生成库**，只是**自己实现词表生成**的过程中，主要调用了`sklearn.cluster`中的`KMeans`进行K均值聚类。实现如下：

```
def clusterDescriptors(descriptors, no_clusters):
    kmeans = KMeans(n_clusters=no_clusters).fit(descriptors)
    return kmeans
```

聚类的数量也即词表大小由变量`num_clusters`确定。

3.3 图像特征提取

这部分实现用词表来表示图像，也即统计图像特征在词表中各特征的匹配数量，用`kmeans`对图像的每个sift特征进行分类预测，统计各类的频率，此后还需进行频率的归一化，在main.py中实现如下。

```
def extractFeatures(kmeans, descriptor_list, image_count, num_clusters):
    im_features = np.array([np.zeros(num_clusters) for i in range(image_count)])
    for i in range(image_count):
        for j in range(len(descriptor_list[i])):
            feature = descriptor_list[i][j]
            feature = feature.reshape(1, 128)
            idx = kmeans.predict(feature)
            im_features[i][idx] += 1
    return im_features


def normalizeFeatures(scale, features):
    return scale.transform(features)
```

3.4 分类器设计

本次实验选用的是SVM分类器，调用`sklearn.svm`中的`SVC`，分类器的定义、拟合函数中主要的部分如下：

```
def findSVM(im_features, train_labels):
    features = im_features
    params = # 设计参数
    svm = SVC(params)
    svm.fit(features, train_labels)

    return svm
```

3.5 训练和测试模型

在main.py的主程序中，程序运行的主干就是两个函数：模型训练函数`trainModel()`和测试函数`testModel()`。`trainModel()`主要工作流程为：读取训练图像，通过文件名得到其类别，保留所需类别（即0，2，4，6，8，其实**理论上这个程序是可以实现10类图片的分类的，但是运行时间会非常久**）的训练样本，对所有样本进行sift特征提取，然后在所有特征描述符空间进行K均值聚类生成视觉特征此表，然后用词表频率描述每个训练样本，最后经过SVM训练获得分类模型。

`testModel()`接收`trainModel()`获得的K均值聚类模型（词表）、分类器（svm）和词频归一化尺度信息scale（是一个`sklearn.preprocessing.StandardScaler`对象），完成如下工作：获取测试样

本及类别，保留所选类别，进行sift特征提取，执行`extractFeatures`获得图像的词表表示，然后用SVM分类器进行分类预测，最后调用函数，计算并输出结果。

3.6 结果输出相关功能

以下是几个有关结果输出的函数：

```
def plotConfusionMatrix(y_true, y_pred, classes, normalize=False, title=None,
cmap=plt.cm.Blues)
```

此函数用于统计每个类别的准确率并绘制混淆矩阵（调用`sklearn.metrics.confusion_matrix`）。

```
def plotConfusions(true, predictions)
```

此函数接受真实类别标签（0-4数字）和预测类别标签列表，进行预处理（主要是获取类别名），然后调用`plotConfusionMatrix`绘制混淆矩阵。

```
def plotHistogram(im_features, num_clusters)
```

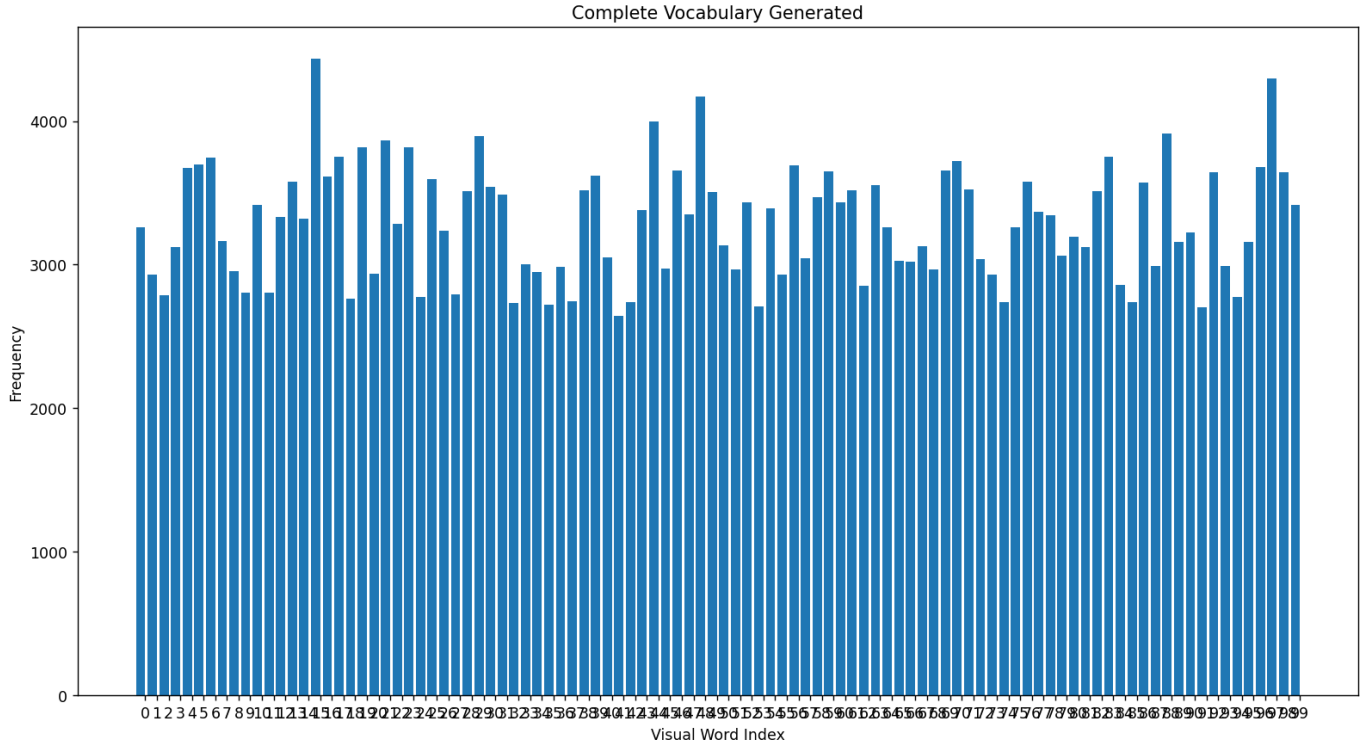
此函数统计词表中每个特征在训练集的所有图像中出现频次，并进行直方图可视化。

```
def findAccuracy(true, predictions):
    print('accuracy score: %0.3f' % accuracy_score(true, predictions))
```

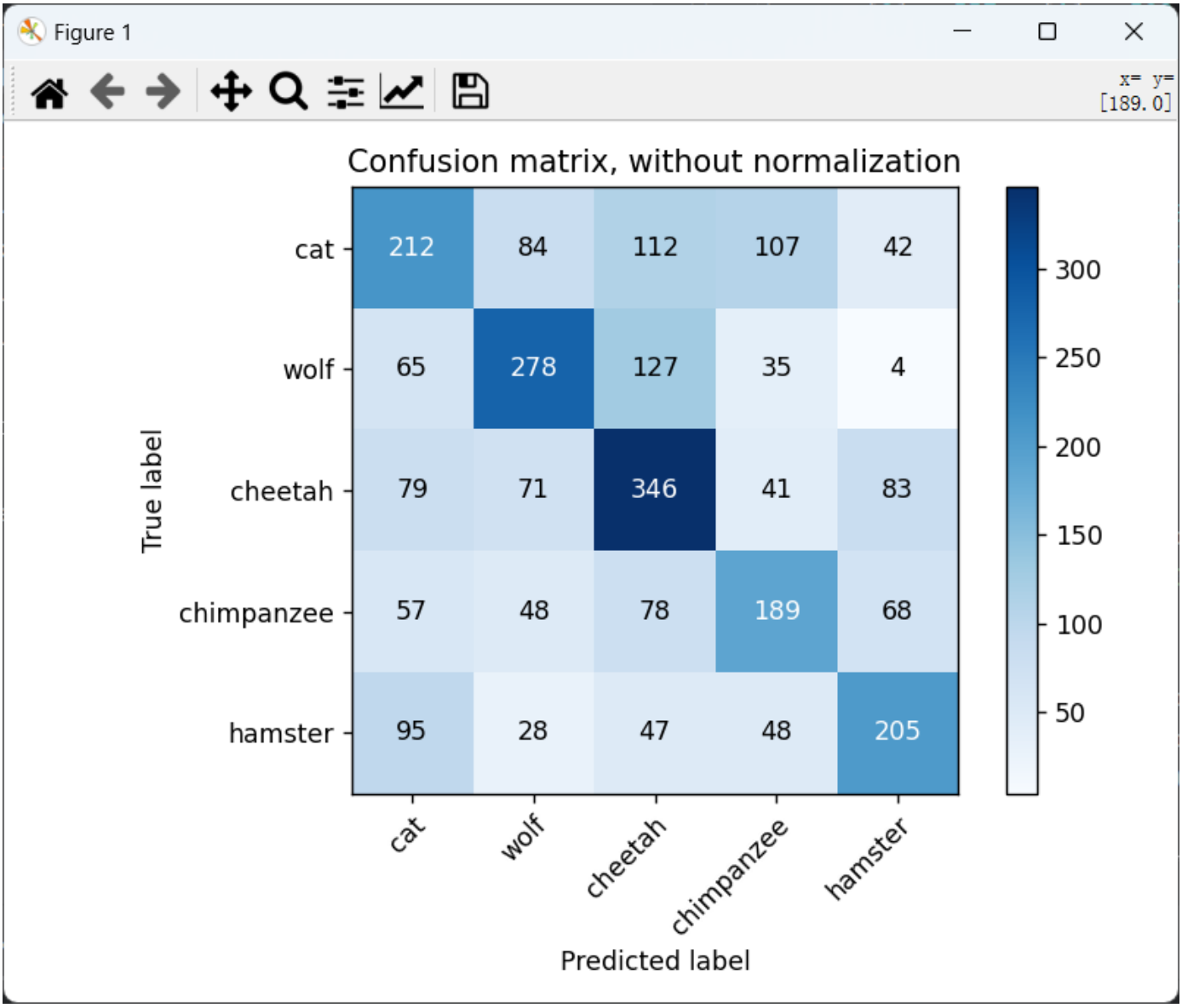
调用`sklearn.metrics.accuracy_score`进行最终测试准确率的计算和输出。

4. 实验结果

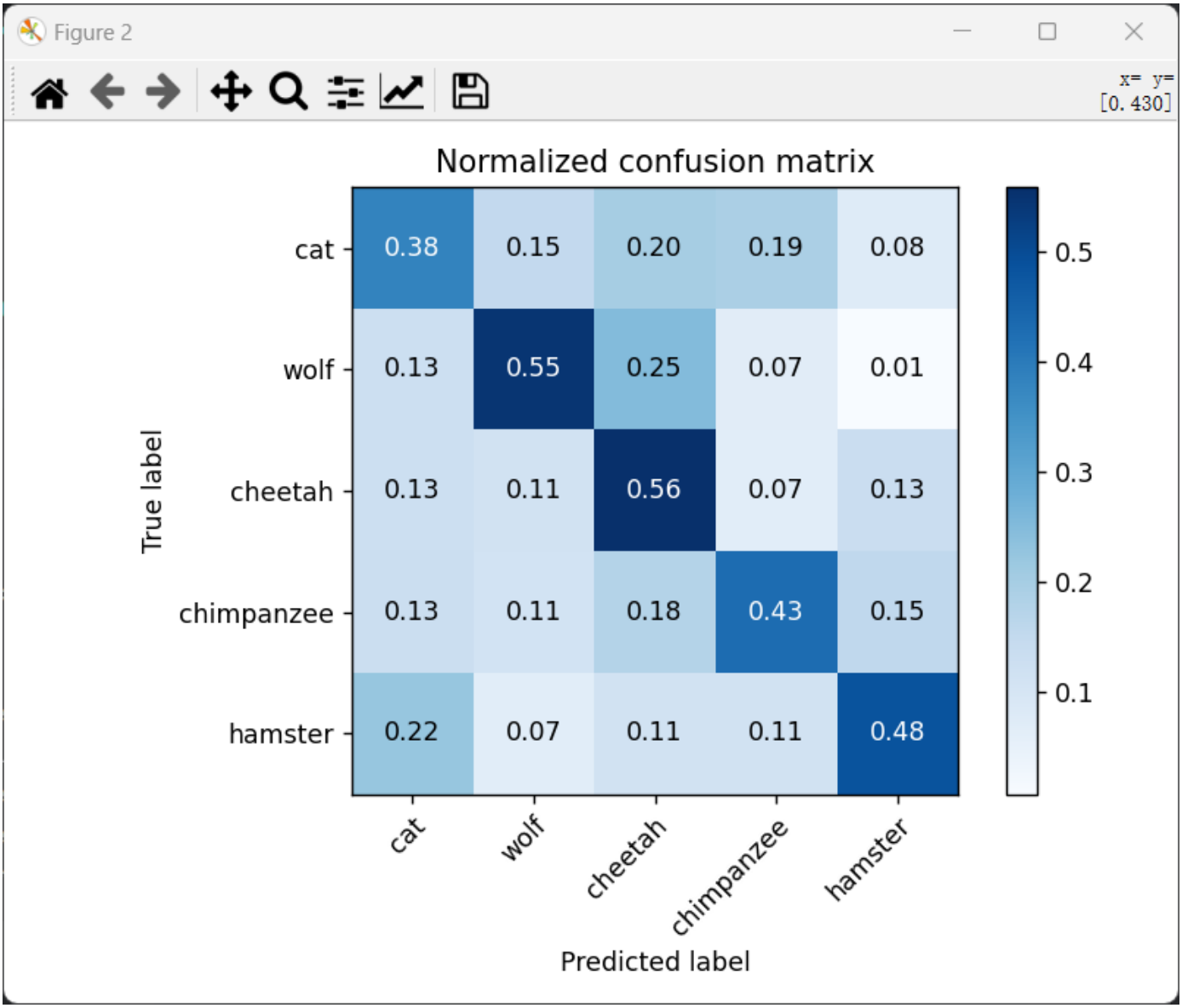
实验中取词表大小`num_clusters=100`，训练集上词表频率直方图如下：



测试集上进行分类测试的结果混淆矩阵，对角线即为正确分类的个数：



归一化后的混淆矩阵：



最终输出的准确率得分：

```
accuracy score: 0.483
```